第1章

組み込みソフトウェア開発の効率の良い学びかた

――組込みスキル標準(ETSS)からコーチングまで

山本雅基

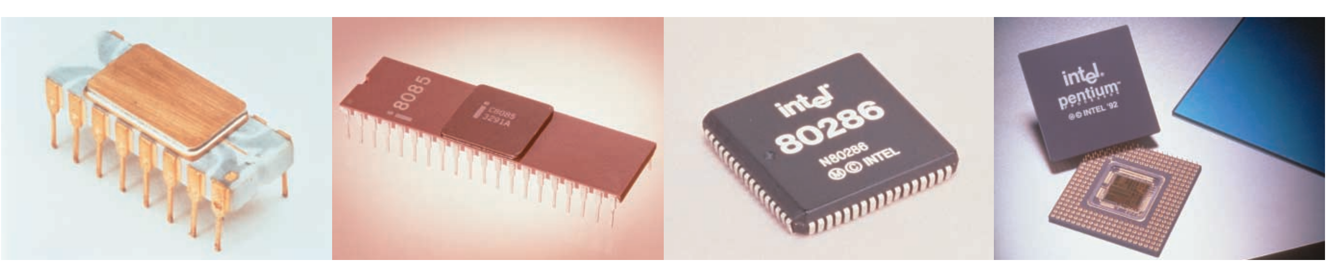
ここでは組み込みシステム開発に必要なスキルを効率良く習得する方法を紹介する．組み込みソフトウェア開発に必要なスキルを体系化したものとして，独立行政法人 情報処理推進機構 ソフトウエア・エンジニアリング・センター(IPA/SEC)が作成した「組込みスキル標準(ETSS)」がある．これに基づいて自身のしごとを分析すると，組み込み技術の全体像やスキルアップの手順が見えてくる．（編集部）

組み込みソフトウェア技術にかかわるしごとをされている皆さん，いかがお過ごしでしょうか．日々，しごとに追い立てられていませんか．中には，「このままで良いのか」と，将来を不安に思われている方もいらっしゃるのではないでしょうか．

忙しい日々，不安に襲われる日々，どれも筆者が経験してきた日々です．1980年代の初めから組み込みソフトウェア技術者であった筆者は今，「組み込みソフトウェア技術者教育」という課題に取り組んでいます．技術者を教育するという立場から皆さんのお役に立ちたいと思い，この稿を執筆しました．筆者は，本業では教育カリキュラムや教材の開発と講義を行っていますが，本稿では，個々の教材の話ではなく，より上位の視点から組み込みソフトウェアの学びかたについて解説します．

現在，「組込みスキル標準(ETSS)」が注目を集めています．ETSSは，これからの組み込みソフトウェア技術者の一般常識になると思います．ここでは，技術者個人でETSSを活用する方法を紹介します．

さらに，これまた現在注目されている「コーチング」についても紹介します．コーチングとは，スポーツの世界の話ではありません．最近では，大型書店に行くと，ビジネス書の棚にコーチング関連の書籍が多数並べられています．Googleでコーチングを検索すると，200万件以上ヒットします(2006年3月現在)．コーチングは，ビジネスの世界で大流行しています．ここでは，自分で自分をコーチングするセルフコーチングの概要について述べます．きっと，皆



(a) 4004(4ビット，1971年)　(b) 8085(8ビット，1976年)　(c) 80286(16ビット，1982年)　(d) Pentium(32ビット，1993年)

写真1　マイコンの移り変わり

最初のマイコンである4004は，嶋正利氏(ビジコン)とTed Hoff氏(米国Intel社)によって開発され，1971年に発表された．どの時代であってもソフトウェア技術者は，与えられた制約(メモリ容量，CPU性能など)の中で最大限の努力を行っている．

KeyWord　組込みスキル標準，ETSS，コーチング，IPA，スキル基準フレームワーク，マイコン，セルフコーチング，カー・ナビゲーション・システム，アセンブリ言語，能力開発

さんのしごとの役に立つと思います.

新年度が始まりました.新しいことを始めるにはとても良い時期です.さぁ,初心に帰って,元気に勉強していきましょう.

1 行き当たりばったりの能力開発

まず,四半世紀も前から組み込みソフトウェア技術者だった筆者が,どのようにして組み込みソフトウェアの技術を身につけていったのかを紹介します.振り返ってみると,かなり行き当たりばったりの行動だったと思います.組み込みソフトウェア技術者としての成長という観点では,これはかならずしも良い例とは言えないかもしれません.

● 社会人生活の始まりは不安だらけ

少し,昔話にお付き合いください.1960年より前に生まれた方は,懐かしみながらお読みください.また,若い皆さんは,想像を膨らませながらお読みください.

1980年代初めのお話です.当時,会社にはワープロや電子メールはありませんでした.複写機は高価で自由に使えず,青焼きでした.あの感光紙の手触りを今も思い出します.「マイコン(**写真1**)プログラミング」とは言っていましたが,「組み込みソフトウェア」ということばを聞いた記憶はありません.携帯電話はもちろんありません.カー・ナビゲーション・システムは,それらしいものを007のジェームス・ボンドが映画の中で使っていました.今の組み込み機器のほとんどが,当時は存在しませんでした.組み込みシステムの技術は,まだれい明期にありました.

学校を卒業した筆者は,下宿を引き払い,Uターン就職しました.社会人としての新しい生活が始まったのです.朝6時に起床したのは何年ぶりだったでしょう.着慣れぬスーツを着て,1時間半の電車通勤でした.社員証には少し緊張した自分の顔写真が貼られていました.自分の机の引き出しには,まだ何も入っていません.顔を上げれば,いつも打ち合わせをしている上司や忙しそうにしている先輩エンジニア,緊張している同僚の顔が見えます.

筆者は少しずつ,不安になってきました.実は,学校ではマイコンなど触ったことがありません.どうやら,しごとではアセンブリ言語のプログラムで機械制御を行っているようですが,筆者はFortranで統計計算しか行ったことがありません.「すんなりとしごとに入っていけるのだろうか…」.筆者は,入社当初から不安に襲われていました.

● 習うより慣れろ

不安を打ち払いたければ,しごとに慣れればよいのです.筆者の当時のしごとは,アセンブリ言語でプログラムを書き,実機でデバッグすることでした.読んだことのなかった英語のCPUマニュアルを読み,アセンブリ言語を勉強し,使ったことがなかったミニ・コンピュータ(エンジニアリング計算などに使う小型コンピュータ)のコンソールを使い,存在すら知らなかったROMライタでROMを焼きました.

すべてが試行錯誤でした.勉強しようと思っても,教科書なんてありません.ただひたすら,先輩の書いた謎めいたアセンブリ・コードを読み,朝から夜までしごとに明け暮れていました.すっかりプログラムを暗記し,頭の中で必死にデバッグしていました.

「あぁ,きょうも動かなかった」.落胆して帰宅する途中,駅のロータリを横切ろうとしたときに頭の中で急にバグを発見しました.お風呂に入っているときに,頭の中でもっとコンパクトな(少メモリ容量の)実装方法を見つけました.今でも,そのときの感動をきのうのことのように思い出します.楽しかったです.ほんとうに,楽しい日々でした.きのうまでできなかったことが,きょうはできるようになっている.しごとは,ほんとうに楽しい!

筆者は,すっかりりっぱな6801(米国Motorola社が開発した8ビット・マイコン)のアセンブリ・プログラマになっていました.知らないうちに,しごとへの不安感は薄れていきました.



1981年2月号
特集「最新版・6809応用テクニック」
- 位置独立型の6809セルフアセンブラ
- 6809ソフトウェア開発システム
- 6809用基本モニタ・プログラム
- 6809ファミリLSI最新情報
- ステップ動作をソフトウェアで制御する方法

図1 組み込みソフトウェアれい明期の勉強(インターフェース,1981年2月号)

今も昔も,勉強熱心な技術者は,さまざまな手段で勉強している.当時,組み込み関連の雑誌は種類が少なかったのだが,最近では選択に迷うほど多くの雑誌が出版され,ずいぶんと勉強しやすい環境になっている.なお,1981年2月号はすでに絶版になっており,入手できない.

● そうだ，勉強しよう

しごとに慣れてくると，欲が出てきます．「もっと新しい技術に触れたい」という欲求です．当時，マイコンは急速に発達していました．アセンブリ言語や割り込み処理，A-D変換，D-A変換など，しごとで使う技術をもう少しきちんと勉強したいと思いました．

しかし，ここでハタと困りました．社外のセミナは充実しておらず，課内で行われていた勉強会が唯一の教育の機会でした．そもそも，勉強するための教科書や参考書も，それほど数があるわけではありません．

かろうじて，CQ出版の「インターフェース」など，2～3の技術誌にマイコンやソフトウェアの技術解説が書かれている程度です（**図1**）．これらを，むさぼるように読んでいました．

● 習うより慣れろ，再び

しごとにかなり慣れてきたと思っていた矢先に，割り当てられていたしごとの内容が変わりました．それまでは機械の制御を行っていたのですが，新しいしごとでは画像表示装置への描画を行うことになりました．CPUは8085（米国Intel社が開発した8ビット・マイコン）になり，アセンブリ言語（CPUごとに異なる）をもう一度覚え直さなければなりません．取り扱う周辺チップも変わりました．当時は高機能なグラフィックスLSIはなく，出始めたゲーム用のICを試す日々が始まりました．サブルーチンも最初から作り直し，ファイル構成も変えました．

再び習うより慣れろです．筆者はすぐに，すっかりりっぱな8085アセンブリ・プログラマになり，描画ライブラリ（多数の描画用ソフトウェア部品の集合体）を作るに至りました．

● 勉強，再び

描画ライブラリを使い始めると，CPUの性能が低いため，描画に時間がかかりすぎることが問題となりました．そこで，描画アルゴリズム（負荷の軽い描画の処理手順）について勉強しようと思い立ち，いろいろな資料を読みあさりました．勉強を通じて筆者は，直線や円弧を描くためのBresenhamアルゴリズム(1)を知り，8085のアセンブリ言語で実装し，高速描画が可能なライブラリを作りました．描画処理に乗算を使用していた従来の実装と比べて，劇的に描画速度が向上しました．勉強した技術を即座にしごとに生かすことができて，感動しました．

描画ライブラリには塗りつぶしなどの機能も用意する必要があるため，その後も描画アルゴリズムの勉強を続けました．この業界の進歩は速いため，いつも教科書や参考書には苦労しますが，それは誇らしい苦労です．教科書にもなっていないことをしごとでしているのですから….

● 行き当たりばったりはもういやだ！

実は，その後もさまざまなしごとを担当しましたが，能力開発は同じように行ってきました．つまり，会社の事業戦略に基づく上司の指示で急にしごとが変わり，しごとを通じてのスキルの開発を行い，それに飽きたらずに勉強を行う，この繰り返しです．右も左もわからず，言われたしごとをこなしながら勉強を続けてきました（**図2**）．

これが，組み込みソフトウェア技術者の能力開発の一つの道です．少なくとも筆者は，管理職になるまでは，この道を歩いてきました．でも，この道は"正しい道"でしょうか．

会社の事業戦略によって急にしごとの内容が変わることは，会社員であるかぎり，受け入れなければなりません．ただし，その後のスキルの開発と勉強が，あまりにも場当

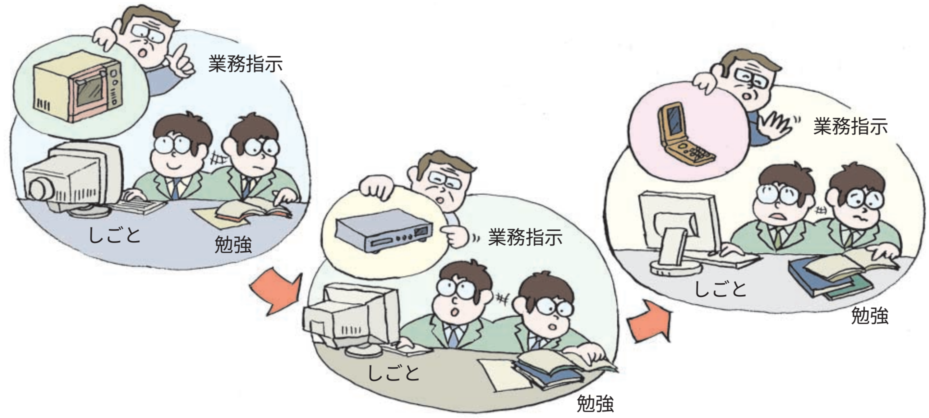


図2
右も左もわからずしごとをこなして勉強を続けた筆者

会社員なので，「上司の指示でいろいろなしごとに従事すること」は受け入れるし，楽しいことでもある．しかし，組み込みソフトウェア技術者としての普遍的な知識やスキルをきちんと理解していなかったため，しごとの内容が変わるたびに，徐々に自分を見失うようになっていった．

たり的過ぎたのではないかと思います．筆者はときどき自分を見失い，不安に思う日々を過ごしていました．

そもそも，「組み込みソフトウェア技術者」としてのアイデンティティや誇りはどこにあるのでしょう．悩みました．もし，組み込みソフトウェア技術者としての自己の役割をきちんと把握していれば，アイデンティティを確立でき，もっと元気にしごとや勉強に取り組めたのではないかと思います．

行き当たりばったりはもうたくさんです．もっと見通し良く組み込みソフトウェア技術者として成長する道はないのでしょうか．もし，そういう道があるのなら，若い皆さんには，ぜひその道を歩いてほしいのです．さあ，探してみましょう．

2 何を勉強すればよいかわかっていますか？

そもそも組み込みソフトウェア技術者は，産業界でどのように思われているのでしょう．そして，学校や企業ではどのように教育されているのでしょう．個人として，何を勉強していけばよいのでしょう．

● 産業界が組み込みソフトウェア技術者を求めている

組み込みシステムとは，電子機器に組み込まれ，その機器のハードウェアを制御するコンピュータ・システムのことを言います．組み込みシステム関連産業は，日本においてはたいへん重要な存在です．身の回りの製品では，自動車や携帯電話など，組み込みシステムによって競争力を高めている製品が多数あります．独立行政法人 情報処理推進機構（IPA）は，「2005年版 組込みソフトウェア産業実態調査報告書[(2)]」において，次の統計データを示しています．

- **組み込みシステム関連産業がGDP（国内総生産）に占める割合は10.2%**
- **製品開発費用の中で，組み込みソフトウェアの開発費用は全体の49.5%を占める**
- **組み込みシステム開発に従事している技術者の46.3%（17.5万人）が組み込みソフトウェア技術者**
- **7万人の組み込みソフトウェア技術者が不足している**

つまり，日本の組み込みシステムが適用された製品は，日本経済の競争力の原点になっているのです．そして，中でも私たち組み込みソフトウェア技術者が果たす役割は，年々大きくなっています．私たちは必要とされ，さらに多くの仲間を産業界は求めています．縮小均衡の産業が多いなか，喜ばしいことだと思います．

● 組み込み関連の学校教育は始まったばかり

マイコンが初めて発売されたのが1971年であり，1970年代には日本の大学に情報系の学科ができました．2006年の時点で，学校において，基本的なソフトウェアの教育はずいぶんと行われるようになってきました．しかし，組み込み関連の教育はまだ始まったばかりです．

皆さんは学生時代，組み込みソフトウェアに関してどのような教育を受けてきましたか？ ある方は，プログラミング実習の単位を取られたでしょう．では，その実習の中で，マイコンのタイマ割り込みプログラミングを行った方はどれだけいるのでしょう．LinuxのC++プログラミングしか経験しなかった方のほうが多いのではないでしょうか．また，ある方は，ソフトウェア工学の講義の中で構造化設計を学ばれたかもしれません．しかし，どれだけの方が暗記知識としてではなく，実際に組み込みシステムを構造化設計によって開発できる能力を身につけているのでしょうか．

皆さんは自身の経験から，自分が受けた，ともすれば教条主義的な学校教育が不十分であったと思われていることでしょう．しかし，すでにその問題点は認識されており，現在，学校ではさまざまな取り組みがなされています．近い将来，この問題点は解消すると期待されています．

● 企業経営者は当事者の視点で考えることはできない

組み込みシステム開発費の中でももっとも費用がかかっているソフトウェア開発費を圧縮するためには，ソフトウェアの生産性（開発効率）や品質を向上する必要があります．利益追求の点から考えても，企業は組み込みソフトウェア技術者の教育を推進するべきです．

しかし，経済産業省の調査によると，企業は十分な時間の社員教育を行っていないという実態が明らかになっています（**図3**）[(3)]．このままでは人材は育たず，ソフトウェア開発費は増大の一途をたどり，企業の存続と発展に支障をきたすのではないかと心配になります．企業の事業責任者は，教育を行う際の課題として，忙しくて時間がとれないことを最大の理由として挙げています（**図4**）[(3)]．中長期を見据えて戦略的に人材育成を行う取り組み意識が弱いように感じられます．

企業の経営者や事業責任者は，実際に自身で組み込みソ

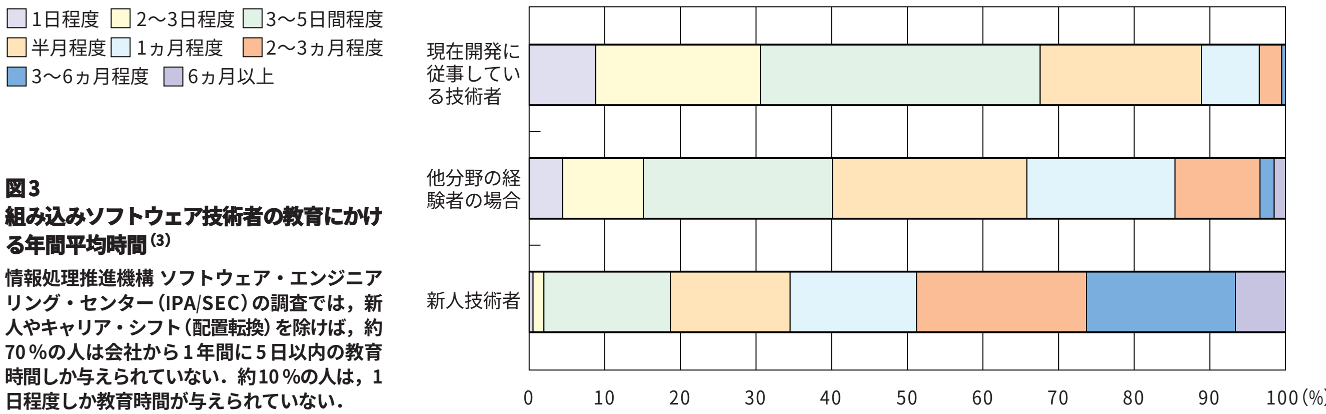


図3
組み込みソフトウェア技術者の教育にかける年間平均時間(3)

情報処理推進機構 ソフトウェア・エンジニアリング・センター(IPA/SEC)の調査では，新人やキャリア・シフト(配置転換)を除けば，約70％の人は会社から1年間に5日以内の教育時間しか与えられていない．約10％の人は，1日程度しか教育時間が与えられていない．

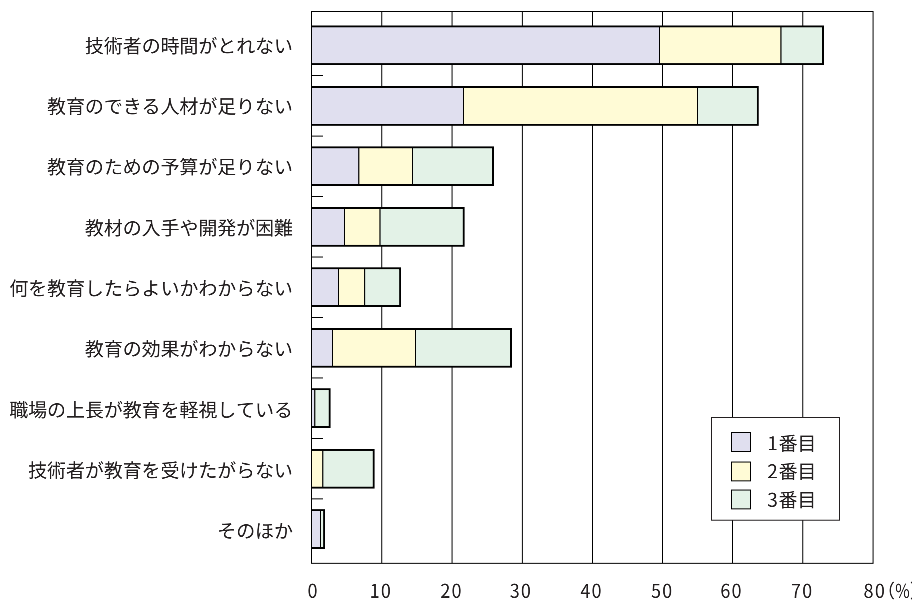


図4
組み込みソフトウェア技術者の教育を行う際の課題(3)

IPA/SECの調査では，事業責任者は，技術者の時間が取れないことを，教育を行う際の最大の課題として挙げている．技術者に教育時間を与えていないのは，はたしてだれだろうか．事業責任者は，教育時間を与えようと考えているのだろうか．

フトウェアの勉強を行うわけではありません．ですから，教育を受ける当事者の視点で考えることはできません．勉強するのは，事業責任者ではなく皆さんです．皆さんの目線で考える必要があります．

では，経営者が教育のための時間を確保するように指示すれば，組み込みソフトウェア技術者教育が着実に行われるようになるのでしょうか．ひょっとしたら，時間の不足以外にも隠された原因があり，時間が十分にあっても企業教育は進展しないのではないでしょうか．問題の本質に迫るためには，等身大の自分の問題として，開発現場の目線で考えることが必要です．事件(!?)は会議室で起こっているのではなく，開発現場で起こっているのです．産業や企業の視点ではなく，あなた自身の視点で考えてみましょう．

● 何を勉強すればよいのかわからない自分に気づく

かりに自分がフリーランスの(特定の会社に属さない)組み込み技術者である，と考えてみてください．フリーランスの技術者であるあなたは，経営者であり事業責任者でもあります．そのうえで，**図4**で示した企業に対するアンケート項目にあなた自身が答えてください．もちろん自分を高く売り込むために知識獲得とスキル向上が必要ですから，勉強する時間はかならずねん出することでしょう(あなたがいちばん勉強していたころのことを思い出して！)．

いかがでしょう．勉強を行う際の課題はなく，どんどん勉強している自分をイメージできましたか？ それとも，時間以外の課題に気がつきましたか？

実は，多くの方は，組み込みソフトウェアについて勉強しようと考えたとき，何を勉強すればよいのかわからない

自分に気づき，がく然とします．教えてくれる人がいない，という課題を思いついた方も，実際には何を勉強するかは教える人任せで，自分では深く考えていない方が多いのではないでしょうか．

このように，「組み込みソフトウェア技術者として，何を勉強していけばよいのかわからない」ということが，組み込みソフトウェア技術者教育のもっとも根源的な問題になっています．

3 スキル基準による“見える”化

では，いったい「組み込みソフトウェア技術」には，どのようなものがあるのでしょうか．さまざまな分類がありますが，ここでは組込みスキル標準（ETSS）で期待されるスキル基準を取り上げることにします[(4),(5)]．ETSSは，組み込みソフトウェア開発力強化のための人材育成と人材活用の実現を図るために定められており，社会人の皆さんにとっては有効かつ現実的です．

公開されているスキル基準Ver.1.0を活用するだけでも，十分な効果があります．スキル基準は，いわば広大な組み込みソフトウェア技術という海に乗り出すための海図のようなものです．この海図があれば，組み込みソフトウェア技術者として何を勉強すればよいかがわかり，見通し良く勉強に取り組むことができます．もちろん，現在ドラフトであるキャリア基準と教育カリキュラムのVer.1.0が公開されれば，海図の上にキャリアという航路が引かれ，教育カリキュラムという港が書き込まれることになるので，よりいっそう皆さんの技術力の向上に役立つと思います（図5）．

スキル基準の詳細については，参考文献(4)，(5)や，本誌2005年7月号，pp.103-112の解説「『組込みスキル標準』を活用しよう」をご覧ください．ここでは，独学で勉強するうえで，スキル基準をどのように使えばよいかのヒントを示します．

● スキル基準フレームワークを理解する

スキル基準では，組み込みソフトウェア技術を整理する際に「スキル基準フレームワーク」というものを使用します．スキル基準フレームワークの例を図6に示します．フレームワークは，以下の三つの要素から構成されます．

- **スキル・カテゴリ** ── スキルのカテゴリ分けを表す
- **スキル粒度** ── スキルのカテゴリの深度を表す
- **スキル・レベル** ── スキルのレベルを表現する



図5　組み込みソフトウェア技術者として見通し良く技術を勉強する
組み込みソフトウェア技術者として見通し良く技術を勉強していくために，ETSSの「スキル基準」，「キャリア基準」，「教育カリキュラム」の3点セットがある．これは右も左もわからない海を渡るときに頼りになる海図のようなものである．海図は，あなたの場所や進むべき海路，燃料を補給する港を示してくれる．

スキル基準フレームワークでは，スキル・カテゴリとスキル粒度を用いて，組み込みソフトウェア技術を分類します．そのうえで，各組み込みソフトウェア技術に対して，スキル・レベルを位置づけます．

広大な組み込みソフトウェア技術を整理する切り口として，最初に着目すべきはスキル・カテゴリです．スキル基準は，次の三つのスキル・カテゴリで構成されます．

- **技術要素** ── システムに組み込まれ，機能を実現するさまざまな技術要素
- **開発技術** ── 組み込みシステム開発で用いられる技術（手法やツール）
- **管理技術** ── 組み込みシステム開発を円滑に進行させるための管理で用いられる技術（手法やツール）

例えば，カー・ナビゲーション・システムを開発するプロジェクトでは，以下が一例として考えられます．

- **技術要素** ── GPS衛星データから座標を計算する技術
- **開発技術** ── 多様な画面の操作仕様を統一して実現するソフトウェア設計技術
- **管理技術** ── 多くの製品種別へ誤りなく対応する構成管理技術（ファイルのバージョン管理など）

どのような技術であろうと，この3種類に色分けが可能です．そして，このように分類すれば，組み込みソフトウェア開発に必要な技術を整理できます．なお，スキル基準Ver.1.0として，表1のように第2階層のスキル粒度まで定義されているので，参考にしてください[(5)]．

スキル・カテゴリ	スキル粒度				スキル・レベル			
	第1階層	第2階層	第3階層	項　目	初　級	中　級	上　級	最上級
技術要素	情報処理	情報入力	位置情報	GPS				
				地磁気				
				車速パルス				
			地図情報	ナビ研（地図）フォーマット				
				VICSリンク・データ				
		データ処理	地図処理	地図拡大縮小				
				ダイクストラ法（最短経路）				
				近傍情報探索				
	ユーザ・インターフェース	入力装置	ボタン入力	ボタン				
				タッチパネル				
			座標装置	ジョイスティック				
			音声入力	音声認識				
		出力装置	表示出力	LED				
				LCD				
			音声出力	音声合成				
				音源				
開発技術	ソフトウェア・コード作成とテスト	プログラムの作成とプログラム・テスト項目の抽出	コーディング規定	MISRA-C				
				顧客規則				
				部内規則				
管理技術	プロジェクト・マネージメント	タイム・マネージメント	—	パート・チャート				
				ガント・チャート				

図6　スキル基準フレームワークの例

スキル基準フレームワークは，技術とスキルを整理する有益なツールである．これは，あくまでもフレームワークであり，縦方向や横方向にます目を伸ばすことも縮めることも自由．ます目に何を書き込むかも自由である．この図ではスキル粒度としてカー・ナビゲーション・システム開発を想定して例示した．開発技術など，各社の標準を整理し直して使用すると有効性が高まる．

表1　スキル基準Ver.1.0 技術要素スキル・カテゴリ(5)

スキル基準Ver.1.0では，スキル基準フレームワークの第2階層のスキル粒度まで埋められている．これを参考にして，各社・各事業部で，自分たちに適するスキル基準を作る．

第1階層		第2階層		説　明
1	通信	1	有線	WAN，LANなど有線通信技術
		2	無線	電気通信事業用無線，一般業務用無線など無線通信技術
		3	放送	デジタル放送，アナログ放送など放送技術
		4	インターネット	透過的データ転送，アプリケーションなどインターネット通信技術
2	情報処理	1	情報入力	データ入力，音声入力など情報入力技術
		2	セキュリティ	暗号，著作権保護などセキュリティ技術
		3	データ処理	圧縮，データベースなどデータ処理技術
		4	情報出力	マークアップランゲージや文書ビューアなど情報出力技術
3	マルチメディア	1	音声	データ処理，圧縮・伸張など音声処理技術
		2	静止画	データ処理，圧縮・伸張など静止画処理技術
		3	動画	データ処理，圧縮・伸張など動画処理技術
		4	統合	音声・画像などの統合処理技術
4	ユーザインタフェース	1	人間系入力	ボタン，座標など人間系入力デバイス制御技術
		2	人間系出力	表示，音声など人間系出力デバイス制御技術
5	ストレージ	1	メディア	リムーバブル，メモリなどストレージメディア技術
		2	インタフェース	リムーバブル，常時接続型などストレージインタフェース技術
		3	ファイルシステム	ISOやOSネイティブなどファイルシステム技術
6	計測・制御	1	理化学系入力	電気，圧力，光など理化学系入力技術
		2	計測・制御処理	座標・運動，信号処理などの計測・制御技術
		3	理化学系出力	アクチュエータ，光，熱などの理化学系出力技術
7	プラットフォーム	1	プロセッサ	CPU，GPUなどプロセッサ技術
		2	基本ソフトウェア	カーネル，ブートなど基本ソフトウェア技術
		3	支援機能	情報記録，情報収集など支援機能技術

出典：組込みスキル標準2005年版（独立行政法人 情報処理推進機構 ソフトウェア・エンジニアリング・センター）

● 自分のしごとをスキル基準と対応づける

本稿を読んでいる読者の皆さんは，自分のしごとで必要とされている技術を，自身でスキル基準に対応づけることから始めるとよいでしょう．スキル基準フレームワークの表ができれば，現在，自分がどのような技術を求められているのかがひと目でわかります．三つのスキル・カテゴリのどこに入る技術であるかを，自分で表を作りながら意識するだけでも，ずいぶんと技術に対する見かたが深まります．なお，新入社員の皆さんは，スキル・レベルについては深く考える必要はないでしょう．皆さんに要求されるスキルは，「上司の指示のもとで実施できる」というレベルです．

自分の担当業務の分析を行ったうえで，自分が携わっているプロジェクト全体でどのようなスキルが要求されているかを，上司や同僚とともに分析するとよいでしょう．そうすることで，プロジェクトにおける自分の位置づけが明確になるでしょう．開発技術が「プロセス」と対応づけられるので，工程の中における自分の位置づけも理解できます．すると，どこを勉強すればよいかが一目瞭然になり，とても勉強しやすくなります．「先輩はこのスキルを身につけているな」とわかったら，計画的に先回りして勉強しておくこともできます．このように，効率的に勉強できれば，しごとがますます楽しくなるでしょう．そして，しごとや勉強を通じて伸びていく自分のスキルが手にとるようにわかると思います．組み込みソフトウェア技術者としての誇りが自然に生まれます．

● 待たずに自分で始める

ここで肝要なことは，まちがってもよいので，自分でスキル基準への対応づけを行っていくということです．「スキル基準の第3階層に何を書けばよいかわからない」とか，「会社の業務内容がスキル基準Ver.1.0の内容とそぐわないので，改訂されるのを待つ」といったセリフは，単なる言いわけに過ぎません．他人が作った細かく定義されたスキル基準Ver.Xを得たとして，あなたはそのすべてのスキルを獲得するつもりなのでしょうか？ あなたは，どのようなエンジニアになろうとしているのでしょうか？

私たちの時間は有限です．せっかく広大な組み込みソフトウェア技術の海を渡る海図をスキル基準Ver.1.0として得たのですから，その海図に自分たちの海の細かなようすを書き込むことは，あなたの責任で行うべきことです．それとも，混沌とした組み込みソフトウェア技術の海の中で，行き当たりばったりの勉強を続けるのでしょうか？

さぁ，スキル基準によって“見える”化を推し進め，組み込みソフトウェア技術者としてのアイデンティティを確立しましょう．

4 セルフコーチング

スキル基準による“見える”化により，ずいぶんと勉強の見通しが良くなったと思います．しかし問題は，あなたがやる気になって勉強を継続できるか否かです．あなたは，みずから進んでスキルの位置づけを行えますか？ どうしたら，モチベーションを維持して勉強を続けることができると思いますか？

コーチングがそれに対する一つの答えです．コーチングの世界をのぞいてみましょう．

● コーチング vs. ティーチング

コーチングはティーチングとまったく異なります．ティーチングは教え込む教育手法であり，コーチングは能力を引き出す教育手法です．

“ティーチャ”と“コーチ”と言えば，イメージしていただけると思います（p.28のコラム「コーチの語源」を参照）．筆者の子どものころ，教科書片手に一段高い教壇から生徒を見下ろして教えるこわもての中学校の先生（ティーチャ）がいました．会社に入ってからは，つたない筆者のプログラミングを励ましながら見守ってくれた先輩（コーチ）がいました．

このコーチングですが，最近では典型的ティーチャである小中学校の先生も盛んに学んでいます．優れた先生の授業では，単に教科書を読ませるだけではなく，生徒の発言を引き出し，生徒の小さな“気づき”に的確に対応し，生徒に深く理解させる授業が行われています．そこで行われている手法が，まさにコーチングです．今，志の高い小中学校の先生は，独自に勉強会に参加し，コーチングを学んでいるそうです．

● やる気に火をつける

ではなぜ，今，コーチングが盛んになっているのでしょう．一つの理由は，コーチングにより本人のやる気に火をつけることができるからです．勉強では，自主性がもっともたいせつです．くじけそうになったときでもコーチングを適切

に行えば，やる気を燃やし続けることができるのです．

現代は，やる気があるふりをする人はたくさんいます．しかし，本物のやる気を持続できる人は少ないのではないかと思います．だからこそ，いつも本気で困難に立ち向かう，プロフェッショナルのしごとをする人物の生きかたに触れたり，歯を食いしばって練習するオリンピック選手の記録番組を見たりすると，感動するのではないでしょうか．

皆さんは，みずから進んで組み込みソフトウェアの勉強をしようとしているのです．皆さんは，その気持ちを本物にして，やる気を持続していきたいと願っています．優れたコーチがいてくれれば，やり遂げることができるはずです．

しかし，自分の能力を引き出し，やる気に火をつけてくれる優れたコーチはどこにいるのでしょう．

● 自分で自分をコーチングする

しごとにおけるコーチの候補者は，明らかに上司です．しかし，部下は上司を選ぶことができません（でも，逆もまた真であることを，新入社員の皆さんには理解していただきたい）．自分の上司はコーチングなど知らないかもしれませんし，知っていたとしてもコーチング技術がうまくないかもしれません．

そのような不確定な要因に全面的に依存することはお勧めできません．では，どうすればよいのでしょうか．答えは簡単です．自分で自分をコーチングすればよいのです．自分がコーチング技術を学び，自分で自分自身をコーチングするのです．これを「セルフコーチング」と呼びます．

セルフコーチングでは，自分の中にコーチとしてのもうひとりの自分を作ります．そして，自分がコーチとしての自分と対話を行います．これは，いつでもできるし，安上がりだし，とても強力です．もっとも，コーチングの対話技術を勉強しておく必要はあります．とりあえず，1冊くらいコーチングの本を読めばよいでしょう．ほんとうは経験を積むことが必要なのですが，経験はセルフコーチングを通して積んでいきましょう．

もっとも基本的なコーチング的な対話技術は，「よく聞き，受け止めること」です．コーチはしっかり聞き，批判をせずに受け止めるのです．上の空で聞いてもダメです．はぐらかしてもダメです．しっかり，受け止めるのです．その上で，コーチは基本的な質問をすればよいのです．簡単ですが，実はやってみると難しい．しかし，これだけで新しい“気づき”が生まれ，りっぱなコーチングとなります．

セルフコーチングでは，この対話を，自分の中にセルフコーチを作って行います．例えば，次のような対話を心の中で行います．心の中で対話することに慣れていない方は，ワープロなどを使って書きながら対話すると，比較的うまくいきます．書くと悩みがはっきりし，肩の荷が下りる効用もあります．

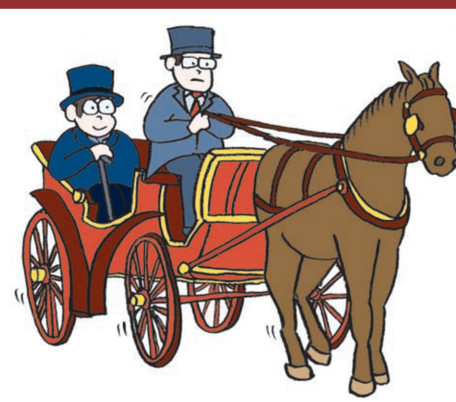
私：「ねぇ，聞いてくれる．勉強がうまくいかないんだ」
もうひとりの私（コーチ）：「そう，勉強しているんだけど，うまくいかないんだ」
私：「うん．時間が取れないんだよ」
もうひとりの私：「そうか，勉強の時間がなくて困っているんだ」

Column　コーチの語源

コーチの語源は，馬車です（図A-1）．今も，4頭立ての4輪大型馬車は「コーチ」と呼ばれています．馬車ができた昔，御者は，乗客の要求を的確に把握し，最適な目的地を設定し，馬車を走らせていました．馬車に乗った乗客は安心です．確実に目的地に到着します．

同じように，コーチは学習者を運ぶ道具です．心の底から学習者のことを思い，学習者を目的地まで運ぶことができる人が，真のコーチです．いばったり，小ばかにしたり，しかったり「だけ」している人は，コーチではありません．そんな馬車には，乗りたくありませんよね．

本稿で紹介したコーチングは，コーチング技術のごく一部です．良いコーチとの出会いは，人生の宝物かもしれません．



図A-1　コーチの語源は「馬車」

私:「ほら，最近，終電までしごとしているでしょ」
もうひとりの私:「そうだね．毎晩遅いね」
私:「そうだよ．納期が来月だからね．せっかくリアルタイムOSの勉強がうまくいっていたのに」
もうひとりの私:「どんなふうに勉強していたんだっけ」
私:「リアルタイムOS上で動く小さなアプリケーションをパソコン上で書いていたんだ」
もうひとりの私:「パソコンを使ってプログラミングしているんだね」
私:「そうだよ．でも時間がなくてプログラミングできないんだ」
もうひとりの私:「そうか．プログラミングする時間がないんだ」
私:「そうだよ．通勤時間帯しか時間がないけど，さすがに電車の中ではできないよ」
もうひとりの私:「何が電車の中でできるかなぁ」
私:「読むことぐらいかな」
もうひとりの私:「読むことはできるんだ．読むことは勉強にならないかなぁ」
私:「ああっ，気づかなかった！ ソース・コードを読む勉強なら電車の中でできるよ」

勉強を継続して行うためには，本気にならなければなりません．本気だと思っていても，さ細なことでつまずき，化けの皮がはがれます．自分が嫌になったり，勉強の継続をあきらめたりすることがあります．そのようなとき，セルフコーチングを行い，再び心に火をつけ直しましょう．

だいじょうぶ．あなたなら，絶対，りっぱな組み込みソフトウェア技術者になれます．絶対にね．

参考・引用*文献

(1) NIST；Bresenham's algorithm，http://www.nist.gov/dads/HTML/bresenham.html
(2)* 経済産業省 独立行政法人 情報処理推進機構；2005年版 組込みソフトウェア産業実態調査 報告書，http://sec.ipa.go.jp/download/200506es.php，2005年．
(3)* 経済産業省 独立行政法人 情報処理推進機構；2005年版 組込みソフトウェア産業実態調査 報告書 ―経営者・事業責任者向け調査―，http://sec.ipa.go.jp/download/200506es.php，2005年．
(4) 情報処理推進機構 ソフトウェア・エンジニアリング・センター；組込みスキル標準ETSS概説書(2005年版)，翔泳社，2005年．
(5) 情報処理推進機構 ソフトウェア・エンジニアリング・センター；組込みソフトウェア開発力強化推進委員会活動報告，http://sec.ipa.go.jp/download/200504eb.php，2005年．

やまもと・まさき
名古屋大学 研究員

1

<筆者プロフィール>
山本雅基．大手自動車部品メーカでの組み込みソフトウェア開発・教育・企画・管理などの経験を積み，2004年から名古屋大学 研究員．社会人向けの組み込みソフトウェア技術者人材養成プログラムNEXCESS(ネクセス，http://www.nexcess.itc.nagoya-u.ac.jp/)を推進．